

YOLO-based object detection performance evaluation for automatic target aimbot in first-person shooter games

Rosa Andrie Asmara¹, Muhammad Rahmat Samudra Anugrah¹, Dimas Wahyu Wibowo¹, Kohei Arai², Mohd Aboobaidur Burhanuddin³, Anik Nur Handayani⁴, Farradila Ayu Damayanti¹

¹Department of Information Technology, State Polytechnic of Malang, Malang, Indonesia

²Department of Information Science, Saga University, Saga, Japan

³Director of UteM International Centre, Faculty of Information and Communication Technology, Universiti Teknikal Malaysia Melaka (UTeM), Melaka, Malaysia

⁴Department of Electrical Engineering, State University of Malang, Malang, Indonesia

Article Info

Article history:

Received Jun 5, 2023

Revised Jan 17, 2024

Accepted Feb 24, 2024

Keywords:

Convolutional neural network

First-person shooter games

Graphical processing unit

Object detection

Real-time detection

You only look once

ABSTRACT

First-person shooter (FPS) focuses on first-person perspective action gameplay, with gunfights usually giving the player a choice of weapons, significantly impacting how the player approaches or strategies. General military-themed FPS games have realistic models with actual weapons' shapes and characteristics. This type of game requires high aiming accuracy while using a mouse on a PC. However, not all players have a fast response time in knowing the surrounding situation. New players may need aid when targeting enemies in the FPS world. One popular yet underhanded method is injecting a program code using a dynamic-link library (DLL) to manipulate memory and asset data from the game. Instead of DLL, we promote a novel approach using the player's real-time game screen, detecting the person without injecting program code into the game. The you only look once (YOLO) algorithm is used as an object detector model since it can process images in real time for up to 45 frames per second. The proposed object detection has an outstanding performance with 65% accuracy, 98% precision, and 61% recall of 51 tests for each game. YOLO's fastest detection speed produces an average of 35 FPS on the YOLO tiny variant using a mixed precision (half) graphics processing unit (GPU).

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Rosa Andrie Asmara

Department of Information Technology, State Polytechnic of Malang

Malang, Indonesia

Email: rosa.andrie@polinema.ac.id

1. INTRODUCTION

First-person shooter (FPS) genre is a first-person perspective video game related to guns and other weapons. FPS often focuses on gunfights, usually giving the player a choice of weapons, which significantly impacts the player's strategies [1]. FPS games usually accommodate realistic models, such as weapon shapes and accuracy, specific rates of fire, and magazines with an actual amount of ammunition [2]. Avatars represent FPS players. Thus, players see the surrounding environment using the avatar's eyes. In FPS, avatars only see arms and properties gripped by the hands. Compared with third-person shooter games or other shooter games, FPS game does not have a camera mode that can change the angle of view to see in a better position. The only way to look around is to move the body in the direction of sight. With this point of view, FPS games are also often used in several other types, such as racing and simulator games, where players act as drivers or pilots in a vehicle. Some examples of popular and early FPS games are Doom, Quake, Deus Ex,

Aliens Vs. Predator, Wolfenstein 3D, and Half-Life. Half-Life became the most influential FPS game ever made [3]–[7].

FPS requires a mouse to aim and shoot the enemy accurately. Nevertheless, novice players face a common problem. They are usually slow while using the mouse to respond to surrounding situations. Some players use aimbot injection to improve aiming and targeting accuracy. This method automatically controls the mouse and keyboard by injecting a program code using a memory-hacking technique with a dynamic-link library (DLL). This program code will manipulate memory and asset data from the game to control computer input devices. Many preventive actions have been published to stop this fraud. One of the most popular methods for detecting the tool hacking game is distinguishing users using the signature-based or heuristic-based memory injection technique [8]. An alternative solution should be established to stop cheating while helping newbies understand the games well.

Object detection computer vision technology has progressed rapidly. Up to ten years ago, object detection was still completed using the extraction feature. This feature is comparatively complicated, especially if the detected object is highly different from the detected object in the training process. Recently, we had you only look once (YOLO), one of the most popular object detections. It is a revolutionary object detection method that uses deep machine learning and non-max suppression, offering the capacity to identify an object within an illustration and its location. Thus, one of the superiorities of YOLO is its ability to detect an object and its location in real time. Consequently, many studies have used YOLO as their object detection tool [9]–[18].

FPS games are a system that runs sequentially at high speed, currently the standard used in FPS games is 60 frames per second. If you use an aimbot, the model and computing of the aimbot will greatly affect the speed and accuracy of detection in the game. Developing an aimbot system using a high accuracy model such as YOLO will increase its ability to shoot targets precisely. The drawbacks of using YOLO need to be taken into account considering that this system must run at high speed.

This study aims to develop a new fair aimbot for an FPS game without hacking memory injection. Instead, we used the natural hacking technique through the game screen shoot technique and enemy detection using YOLO. Further, this system will respond by automatically directing the pointer to shoot into the target enemy, allowing the player to trigger a shoot using the mouse manually. The time detection and aim responses were tested using several YOLO versions, from YOLOV3 to YOLOV6 [19]–[22]. The target shoot accuracy was also evaluated and discussed in the discussion session. For the testing in this study, we used several games, such as Point Blank, Counter-Strike: Global Offensive, Far Cry New Dawn, and Sniper Ghost Warrior Contract 2. Meanwhile, for the screen shoot technique, we constructed low-level library programming using WIN32 application programming interface (WIN32 API), ensuring that the process was performed in real time. From the results of our study, future studies are expected to detect the fraud identified in this research.

This study developed a fair aimbot, which aids game players in completing their mission. However, this development was only completed to analyze the performance of convolutional neural networks (CNNs) object detection, especially YOLO, on real-time cases with relatively high FPS. Several studies have investigated aimbot, but none of them examines the use of CNN (YOLO) as a target aiming at FPS games [8], [23]–[29]. This cheat can be classified as brand-new as it is tough to detect, primarily in the object detection process using an external instrument, such as a camera [24].

Many research articles have discussed the implementation of object detection in games since it is one of the computer vision topics with a quite lengthy discussion. Universally, the recently available object detection is divided into two: object detection based on neural networks and object detection with approaches other than neural networks. Among the object detection with other than the neural networks approach, the most popular ones are template matching [30], feature matching such as orient fast and rotated BRIEF (ORB) [31], scale invariant feature transform (SIFT) [32], speeded up robust features (SURF) [33], histogram of oriented gradients (HoG) [34], and haar cascade classifier [35]–[38]. Template matching is a traditional technique rarely used in recent days with their own library on OpenCV, such as the square difference, normalized square difference, cross correlation, normalized cross correlation cross coefficient, and normalized cross coefficient [39]–[41]. The non-neural network technique was commonly used until ten years ago when neural network-based object detection started progressing.

This study is expected to be a turning point as it offers high accuracy with a robust algorithm representing the global object, not the object detection model with a specific algorithm. The neural network technique presents excellent robustness on uncommon or even partly imperceptible objects, resulting in the increasing popularity of neural network-based object detection. Examples of neural network-based object detection are region-based (cascade R-CNN [42], faster R-CNN [43], fast R-CNN [44], R-CNN [45]), YOLO, single shot multibox detector (SSD) [46], retina-net [47], deformable convolutional networks [48], [49], and single-shot refinedet (refinement neural network for object detection) [50].

2. RESEARCH WORKS

2.1. First-person shooter

3D FPS game is one of the currently popular game genres. This game is complex and fast-paced, especially its multiplayer mode. Accordingly, the players should have relatively excellent and effective navigation skills to go through the 3D environment, target the opponent, and avoid obstacles. This FPS game requires extended learning duration, so a higher playing duration enhances the players' skills and capability.

Two of the earliest FPS video games are Maze War and Spasim. Maze War was initially developed in 1973. One example frame of the Maze War game can be seen in Figure 1(a). Further, the game was subsequently developed at the Massachusetts Institute of Technology, culminating in an eight-player version that could be played through the advanced research project agency network (ARPANET) [51]. Additionally, Doom, Quake, and Counter-Strike mods from Half-Life are examples of successful PC games that substantially influence the global game industry. One example frame of Counter-Strike can be seen in Figure 1(b). Currently, the shooter game is dominated by mobile games with an extensive market, for instance, Doom Android version, PUBG, Call of Duty Mobile, StandOff 2, and so forth [52].

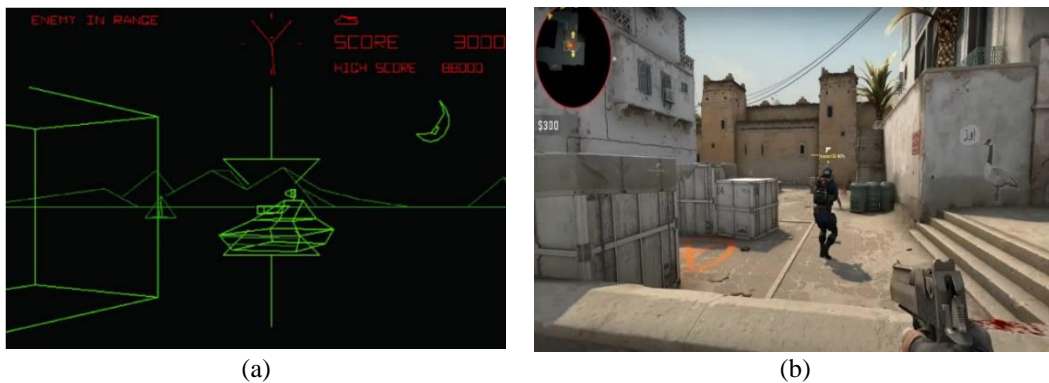


Figure 1. One example frame of the Maze War game; (a) Maze War game and (b) Counter-Strike: Global Offensive

In general, an FPS game is a 3D game where the player plays a character with a specific ability. Mostly, the mission of the FPS game circulates around fighting the opponent team by shooting them. The team that no longer has a member is declared lost, and the next round begins. This cycle is continuously repeated. Further, the games popularity is affected by different weapons being offered, interesting game arena, surprises, team strategy, and team collaboration. Further, the multiplayer feature allows the same game to be played by several players in different places, which also improves the games popularity. This genre has been frequently included in the e-sport team competition. Additionally, FPS games have been occasionally used as war simulations to enhance the war skills of the army troop.

The FPS game is operated by controlling the virtual character by running, walking, and jumping in all directions. The characters collaborate to eliminate the opponent team members rapidly. In PC games, the character can be easily controlled using a keyboard than in the console game, which is regulated using a joystick. Recently, mobile FPS games present the easiest character control through the touch screen. Besides, the character can attain different types of weapons with distinct capacities that can be used to attack the enemy. The accuracy and pace in selecting the weapon become one of the agility parameters in the game. The individual FPS player's skills can be observed from their ability to use the weapon, bullet, and war accessories ownership, accurate movement, precise hiding time, speed of movement, pace in shooting the target, and capacity in navigating direction from the map. Besides, the player's skills are also reflected in their ability to formulate team strategies, positioning the game to be extremely challenging.

2.2. Aimbot

Aimbot is a type of cheating in FPS games that modifies the players infrastructure [26]. One of the most significant challenges in FPS games is directing the pointer into the target opponent using a mouse. This process should be carried out rapidly to lower the opponent's chance of shooting our character. Additionally, the difficulty in directing the pointer increases due to some conditions, such as we can paralyze our opponent faster if we execute the weapon on our target's head.

Figure 2 shows the example of crosshair in Call of Duty: modern warfare 2 games. The crosshair in FPS games usually placed in the middle of the screen, and main mission for the player is to aim crosshair/middle screen for shoot target. The player only has to trigger the shoot, commonly using a left click on the mouse at very high speed. This speed is an important challenge if the player would win the competition. As suggested by its name, aimbot represents the automation of aiming the crosshair pointer at the opponent target without manually moving the mouse. In simple terms, the aimbot detects the target's coordinates and then moves the pointer to the detected coordinate in real time. The pointer-moving process can be completed using I/O device acquisition through a specific algorithm. Meanwhile, the coordinate detection process can be carried out using a number of means related to image processing, such as detecting a human object's movement or presence. Alternatively, we can manipulate the system from the active thread using DLL injection or memory manipulation technique. However, the system manipulation technique can be detected easily. Several methods have been reported capable of detecting this system manipulation, such as the techniques reported in [8], [23], [26]–[29], [53]–[58]. Almost every FPS game has been completed with an auto cheat rejector tool that stops the game whenever the cheat is detected. However, in this study, we used several games that can still be operated using our proposed aimbot.



Figure 2. Call of Duty: modern warfare 2 with a crosshair in the middle

2.3. YOLO object detection

The aimbot development carried out in this study consists of several stages. One of the stages is the target detection process which detects the opponent character. Concerning target detection, computer vision technology, including object detection, has rapidly progressed in the last ten years. Even though artificial intelligence has been widely implemented, great difficulties in recognizing objects are still faced. Additionally, object detection should have been placed in every image area in the previous years, inhibiting its implementation in authentic cases, such as in an autonomous car, video processing, biometrics, or real-time object detection of games. YOLO object detection uses the modified GoogleNet CNN architecture as its image recognition tool. Generally, the architecture of GoogleNet consists of 22 layers in depth or a total of 27 layers in pooling. The network has 100 layers [59] and nine inception modules in linear order. Meanwhile, the network of YOLO consists of 24 convolutional layers, followed by two fully connected layers. Unlike GoogleNet, this network also uses module inception consisting of 1x1 reduction layer and 3x3 convolutional layers [9].

The architecture of YOLO version 3 or YOLOV3 is illustrated in Figure 3(a). Practically, several studies have applied more advanced version of YOLO, from YOLOV4 to YOLOV7 [20], [22], [60], [61], but in this study, we only used YOLOV3 to YOLOV6 since YOLOV7 was just newly published the moment we conducted this study. Further, object detection is completed using unified detection. In this process, a single frame is divided into grid cells, and every grid cell predicts the bounding box of the detected object and provides a confidence score on each box. Besides, the grid cell also predicts the conditional class probability. Then, the predictions are selected based on the class per grid cell probability value to avoid the bounding box getting piled up in the same object, commonly known as the non-max suppression technique [9]. Illustration of detection process using grid cell on YOLO object detection can be seen in Figure 3(b). The algorithm for YOLOV3 is described in the following:

- The input of the image;
- Pass the image through a series of convolutional and max pooling layers to extract features from the image;
- By using the available features, the objects within the image are predicted through a combination of bounding boxes and class probabilities;

- Use non-maximum suppression to remove overlapping bounding boxes and refine the predictions;
- Output in the form of the final set of bounding boxes and class probabilities for each object in the image.

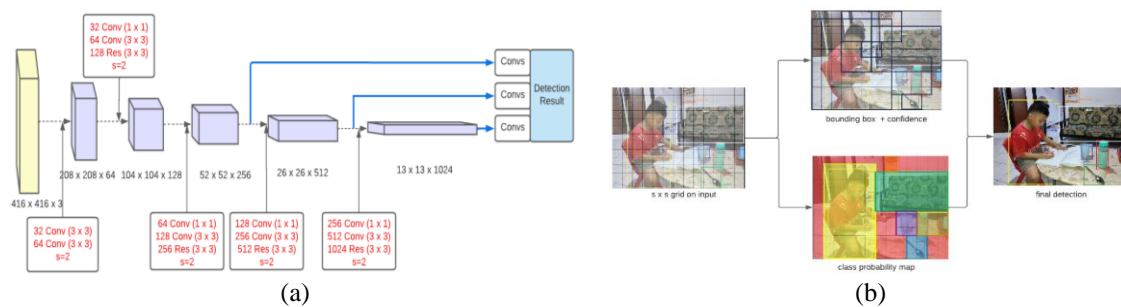


Figure 3. Convolutional block of; (a) YOLOV3 architecture and (b) illustration of detection process using grid cell on YOLO object detection

YOLOV4 and YOLOV5 have been enhanced in a variety of ways, such as in their use of more efficient network architecture and incorporating more techniques like mish activation and ghost normalization to improve their performance. Additionally, YOLOV4 and YOLOV5 also use different approaches to improve object detection accuracy, such as multi-scale predictions and assembling multiple models. The difference in each YOLO version is in their technical specification in which the YOLOV3 uses backbone Darknet53, YOLOV4 uses CSPDarknet53, while YOLOV5 uses focus structure with backbone CSPDarknet53. Focus layer was adopted in YOLOV5 to replace the initial three layers used in YOLOV3. The focusing layer presents some superiorities, such as lower use of compute unified device architecture (CUDA) memory, reduced layer, and enhanced forward and backward propagation [62].

2.4. Screenshot

General games, including FPS, require a quick or real-time response, so the screenshot technique should operate expeditiously. The game with a frame rate of 60 FPS is expected to perform all the processes, from the screenshot to target auto aim, at above 45 FPS frame rate. In this study, we used several types of screenshots, namely Python-multiple screenshots (MSS), PyAutoGUI, PIL, and D3Dshot [63]. MSS is written using ctypes on Python. The module has been integrated into the library Numpy and OpenCV. These modules are the popular tool frequently used in the window screenshot before the recognition process. Additionally, some studies reported that MSS presents faster computation than PyAutoGUI. Thus, we used these four modules in this study.

2.5. Nvidia single and mixed precision

Nvidia single precision is a type of floating-point representation that uses 32 bits to store a numerical value. Single precision is often used in applications that require high performance and low memory usage, such as video games and scientific simulations. In contrast, Nvidia mixed precision uses both single precision and half-precision, which uses only 16 bits per value. Mixed precision allows for a higher level of computational performance by using half-precision for certain parts of a calculation while maintaining the overall accuracy of the calculation by using single precision for the most important parts. This can lead to a significant increase in performance, especially for complex calculations that involve a large amount of data.

3. METHOD

In simple terms, the flowchart block of our system is illustrated in Figure 4. In detail, the processes of our system are described in the following:

- Load model

The initial stage is the load model. The load model was completed using a pre-trained model, which had been trained using common objects in context (COCO) dataset [64] and only adopted the person class. Here are the steps for loading the model; preparing dependencies and library to use graphics processing unit (GPU) acceleration loading model configuration in yolo*.cfg and model weights in yolo*.weights.

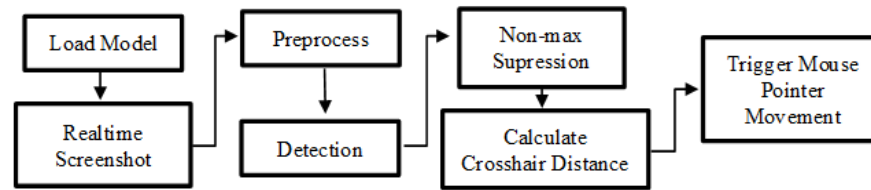


Figure 4. Flowchart of system

– Real-time screenshot

In this stage, screenshot is done in the selected window of the game by attaining the captured frame in real-time to be processed as image input. The image input would be further processed by the YOLO model. In this study, screenshot is implemented using MSS, PyAutoGUI, PIL, and D3Dshot. Here are the steps for real-time screenshot; find target window-capture screen content using Windows graphics device interface (GDI) or Direct3D-accessing frame buffer-compressing image-encode screen content into a specific buffer image format-preview screen content on local screen-adjust frame rate dynamically.

– Pre-process

After the frame was obtained from the screen capture, the image input was resized following the YOLO requirement into 320×320 . The format of image input is then altered into BLOB (object 4D NumpyArray), consisting of images, channel, width, and height. Here are the steps for pre-processing images; resizing image to fit with YOLO requirements (320×240)-ordering channel in RGB colorspace-altered image input into BLOB (4D NumpyArray object).

– Detection

The inference on YOLO was completed in detecting the person object using the input image that had been pre-processed. Then, the program detected the input image. As the system only detected persons, other objects were not detected. This person detection process was completed using pre-trained models YOLOV3, YOLOV4, and YOLOV5, which had been trained using the COCO dataset. Here are the steps for object detection process; initiate list of corresponding bounding box and confidence value-loop every output layer by extracting the id class and confidence value probabilities from every detected object-draw the bounding box.

– Non-max suppression

After that, non-max suppression was carried out to filter the overlap or duplicated detection on every bounding box with intersection over union (IoU) > 0.5 . Here are the steps for non-max suppression process:

- Select the proposed entity box with the highest confidence score, remove it from B (a list of proposal boxes) and add it to the final proposal list D (a list of filtered proposals), initially D is empty
- Compare these proposals with all proposals and calculate the IoU of this proposal with every other proposal. If the IoU is greater than the threshold N, remove the proposal from B
- Take the proposal with the highest confidence from the remaining proposals in B, remove it from B and add it to D
- Calculate the IOU of this proposal with all the proposals in B and eliminate the boxes that have an IOU higher than the threshold value
- Repeat the process until there are no proposals left in B

– Calculate crosshair distance

In this stage, we calculated the distance between the detected objects using the game's crosshair. Crosshair is the center point of the target aim. It never transmigrates and always stays in the middle. To calculate it, we needed to extract the values from the bounding box, namely the x, y, w, and h generated from the YOLO model. Further, the closer distance from the detected person object was calculated by comparing the distance of every detected object. Then, the closest object was selected as the input for trigger mouse movement.

– Trigger mouse movement

After the distance coordinate was obtained, we executed the trigger to simulate the movement by clicking the mouse. This mouse movement input refers to the x, y, w, and h points extracted from the bounding box. The testing was carried out using 51 data frames on every game. Those 51 data were calculated using the confusion matrix and the number of detected FPS numbers. Figures 5(a) to (d) shows the example of the data set for calculating the confusion matrix, precision, recall, and accuracy for Counter Strike, Far Cry New Dawn, Point Blank, and Sniper Ghost Warrior games.



Figure 5. Flowchart of system example of data set for confusion matrix, precision, recall, and accuracy estimation in; (a) Counter Strike, (b) Far Cry New Dawn, (c) Point Blank, and (d) Sniper Ghost Warrior games

4. RESULTS AND DISCUSSION

In this study, the YOLO's network was trained using the COCO dataset, especially the person category. The less clear object detection may be caused by the real person object data used in the COCO dataset training, which used natural clothing (military uniform) without carrying the military properties, such as the weapon. In previous research, creating an aimbot was carried out using memory injection. Memory injection is impossible in multiplayer streaming games because it is automatically detected as a cheat act. This research creates an aimbot from outside the system by screen capturing, specifically using screenshots on the same device, such as MSS, PyAutoGUI, PIL, and D3Dshot. Figure 6 shows 4 FPS games that passed and could use the aimbot screenshot model without being detected as a cheating act.

Figures 6(a) to (d) presents some examples of 3D avatars completed with the general properties and relatively different texture detail in the same resolution. In addition, the graphic detail in the game also highly influences the detected results. In the Counter Strike Global Offensive and Point Blank games as shown in Figures 7(a) and (b) for human characters use closed clothing with less detailed 3D graphics. In contrast, the avatar in Far Cry New Dawn as shown in Figure 6(c) uses open clothing with more detailed 3D graphics with similar resolution as other games. Meanwhile, Sniper Ghost Warrior Contract 2 as shown in Figure 6(d) has a similar texture as Counter-Strike and Point Blank but with more complex 3D graphics.

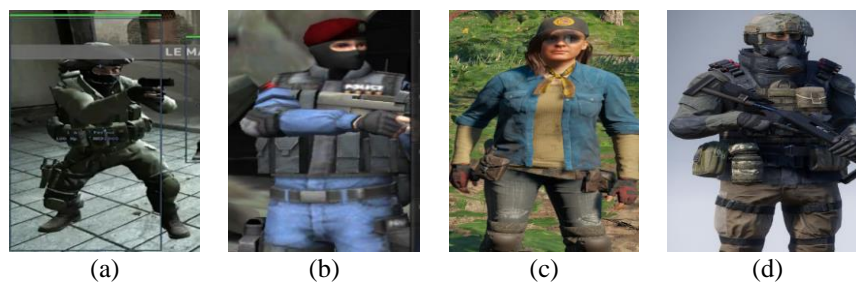


Figure 6. Detail of model person from the different games at 320×240 resolution; (a) CS Go, (b) Point Blank, (c) Far Cry New Dawn, and (d) Sniper Ghost Warrior Contract 2

The results of several testing showed that the 3D human avatar using a military uniform or combat suit completed with the war properties are more challenging to be detected than the character model using casual clothing. Thus, we found a significant difference in detecting the human avatar. The experiment was carried out in real-time using many games. The test was conducted on YOLOV4 with 320×240 half precision game resolution. The object detection test results were relatively excellent, as shown in Figure 7. Figure 7(a)

shows that the object can be significantly detected using the telescopic mode. Meanwhile, Figures 7(b) to (e) show that the object can be detected ideally, even if they are in tiny sizes, such as in Figures 7(b) and (e). Figures 7(c) and (d) also show that the object holding property, such as the long gun, still can be detected precisely, while in Figure 7(f), even if a pillar covers a person's object item, it is still identified correctly.

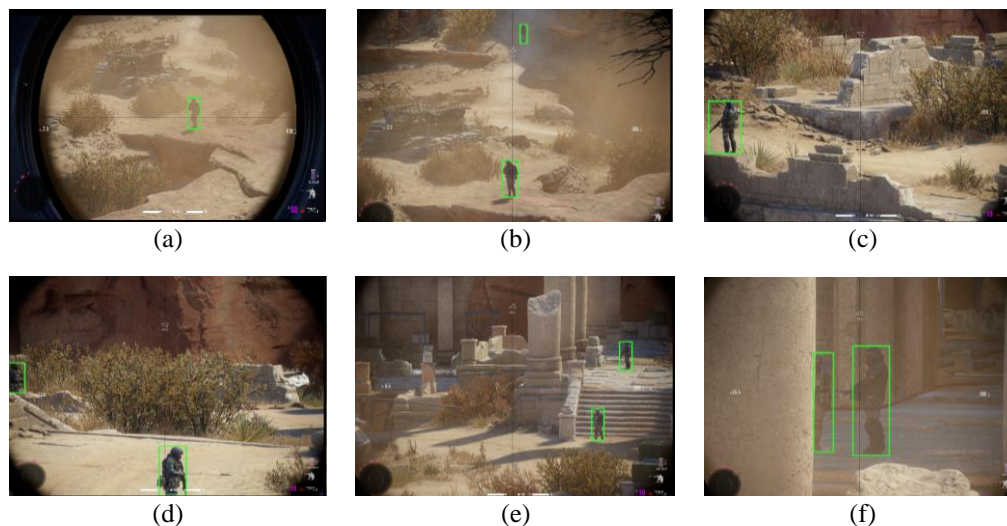


Figure 7. Realtime testing of several daytime scenes in telescopic mode of Sniper Ghost Warrior Contract 2 game; (a) normal object, (b) normal and small object, (c) normal object bring gun, (d) small object bring gun, (e) two small objects, and (f) object with incomplete body captured

Interestingly, the objects wearing relatively thick and complete clothing were still identified accurately. However, during the day, some objects were not detected as their colors were less in contrast with the background color. In some other scenes, objects were always detected so long as they had a highly distinctive color from the background. However, sometimes the system only detected one-third or a quarter of the upper body part of the object. Besides, the object position, whether it faces the front, back, or sides from the camera, presents minimum effects on the results of object detection. The examples of scenes with human avatars in different positions from the camera are shown in Figure 8, suggesting that the object was still detected accurately.

In addition, Figure 8 presents the testing results in real-time during the night scene. The results suggested that almost every object can be detected thoroughly. Figures 8(a) and (e) show excellent object detection, even if the system only detects the person avatars head and some part of its shoulder due to other objects covering the other parts. Besides, the person objects on those Figures also did not face forward, but on the sides. Meanwhile, Figures 8(a) to (f) shows proper object detection, even if the objects are small in size. However, as presented in Figures 8(d) to (f), some character objects were not correctly identified, such as the ones standing behind a truck, above the safety fence, and the airplane wings. These objects are undetected because they stand behind other objects with similar color or in relatively dark places.

The experiment results also suggested that the person object can be clearly detected even in a dark scene, as long as the objects have a contrast color to the background. From the experiment on various games, we obtained similar results accuracy. The detailed accuracy of our results is shown in Table 1.

The experiment results in the night scene showed that YOLO has excellent object detection, even in objects with relatively high noise. YOLO's great object detection capacity is confirmed by its ability to detect 37 from a total of 41 random-person avatars in some different scenes. Besides, we observed relatively similar computation in the day and night scenes, with an average of 15 FPS. This low FPS can be caused by the comparatively high computation from YOLOV4, primarily in the mixed precision. Figure 9 presents the experiment's results conducted in real-time on the day and night scenes using a telescope display and YOLOV4 with different parameters and the same resolution.

The top left of Figure 9 shows the picture obtained using YOLOV4 MP-FP16 in an average of 15 FPS, with the highest and lowest FPS at 12 and 17, respectively. The results showed excellent object detection, even if it appears on a microscopic scale on the screen. However, we do not suggest this FPS value because it is insufficient for the aimbot. Meanwhile, the top right of Figure 9 is the result obtained from using YOLOV4 tiny MP-FP16, with an average FPS of 24, with the lowest and highest FPS of 20 and 26, respectively. This figure shows that the object is not captured correctly. Computationally, the YOLOV4 tiny

MP-FP16 is better than the original YOLOV4, but its results are less accountable to be used in the aimbot. Lastly, the lower part of Figure 9 shows the night scene obtained using the single precision original YOLOV4, with an average 8.5 FPS value. With a meager FPS value, the obtained results have great accuracy, even if only small parts of the object appear within the image.

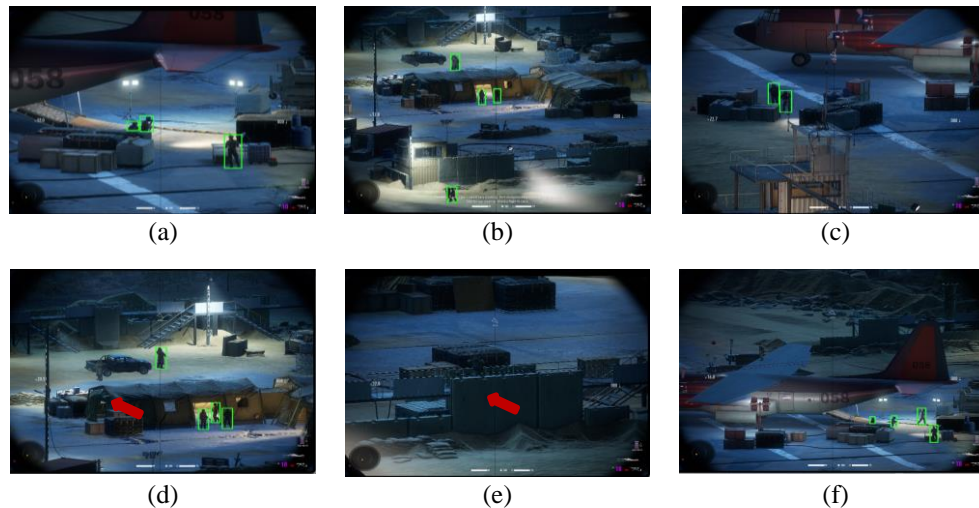


Figure 8. Real-time testing on several night scenes in telescopic mode of Sniper Ghost Warrior Contract 2 game: (a) small incomplete and normal size body, (b) small size bodies perfectly detected, (c) small size body, (d) one undetected body having similar color with background object, (e) one undetected body with darker scene and having similar color with background object, and (f) one undetected with too small size body

Table 1. Results of accuracy, precision, and recall of Point Blank and Far Cry New Dawn from experiment using different types of YOLO

Algorithm	Region	Point Blank			Far Cry New Dawn		
		Accuracy	Precision	Recall	Accuracy	Precision	Recall
YOLOV5S	22×29px	0.568	1	0.488	0.921	1	0.906
YOLOV3 320×320+mixed prec	20×37px	0.313	1	0.222	0.823	0.972	0.813
YOLOV4 320×320+mixed prec	42×101px	0.313	1	0.186	0.725	1	0.674
YOLOV3 tiny+mixed prec	36×103px	0.254	1	0.116	0.49	0.947	0.418
Average		0.435	1	0.337	0.749	0.983	0.711



Figure 9. Real-time experiment on day and night using telescope

In Figure 10, we can observe the obtained results from a similar frame using different versions of YOLO. These frames have profoundly high detection difficulty in which the person's objects are on a microscopic scale, carrying a long-barreled weapon, facing the sides, and in a relatively dark environment close to the edge of the telescopic camera. In both day and night scenes, all the YOLOV4 cannot detect the objects correctly, even the YOLOV4 single precision (top left and top right pictures of Figure 10. Meanwhile, like the previous experiment, the YOLOV4 tiny (bottom right picture of Figure 10 presents

faster computation result with an average of 24 FPS and successful object detection. The YOLOV5s result presented in the bottom right of Figure 10 shows successful object detection but with only 31% accuracy and a better FPS value (17 FPS) than the original YOLOV4.



Figure 10. Comparison of detected results on similar frame using different versions of YOLO

In addition, the YOLOV4 single precision detects an object with 38% accuracy, as shown in Figure 11. This success is caused by the object's location on the outside edge of the telescope's dark area, so it has a relatively higher contrasting color than the background. we also experimented using a confusion matrix consisting of several elements. The true positive (TP) represents the number of objects detected accurately, while the true negative (TN) shows the number of objects not detected accurately.

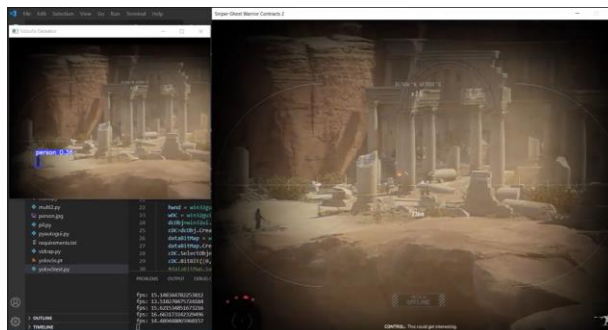


Figure 11. Experiment results using YOLOV4 at 320x240 resolution with object positioned in the center

Additionally, false positives (FP) represent the incorrect detection of an object outside the image, and false negatives (FN) show the incorrect detection of an object within the image. The results of the confusion matrix on 51 experiment data with small-scale selected randomly are shown in Figures 12(a) to (d). In Figure 12(a) can be seen that game Point Blank obtained the best accuracy value of 56.86% using YoloV5s. Figure 12(b) shows that game Counter Strike: Global Offensive got the best accuracy result of 86.27% using YoloV4 320x240+mixed precision. In Figure 12(c) Far Cry New Dawn game got the best accuracy of 92.15% using YoloV5s. Whereas in Figure 12(d) for game Sniper Ghost Warrior Contracts 2 the best accuracy results are 84.31% using YoloV4 320x240+mixed precision. Comprehensively, the obtained accuracy on every game is different, as listed in Table 2. From the experiment using YOLOV5, YOLOV4, and YOLOV3, Point Blank present the lowest object detection than the other games. Meanwhile, Far Cry New Dawn attains the topmost accuracy using YOLOV5.

However, these accuracy results need further analysis. Following the experiment results, the accuracy is affected by several reasons, such as the number of scenes with a similar background and foreground color range, objects detail, the color and texture of the objects clothing, and the objects properties. The results of classification accuracy for all games tested is 0.649, while their precision is 0.987, and their recall is 0.612. The obtained FPS value from the experiment carried out using different YOLO versions and the averages are shown in Table 2.

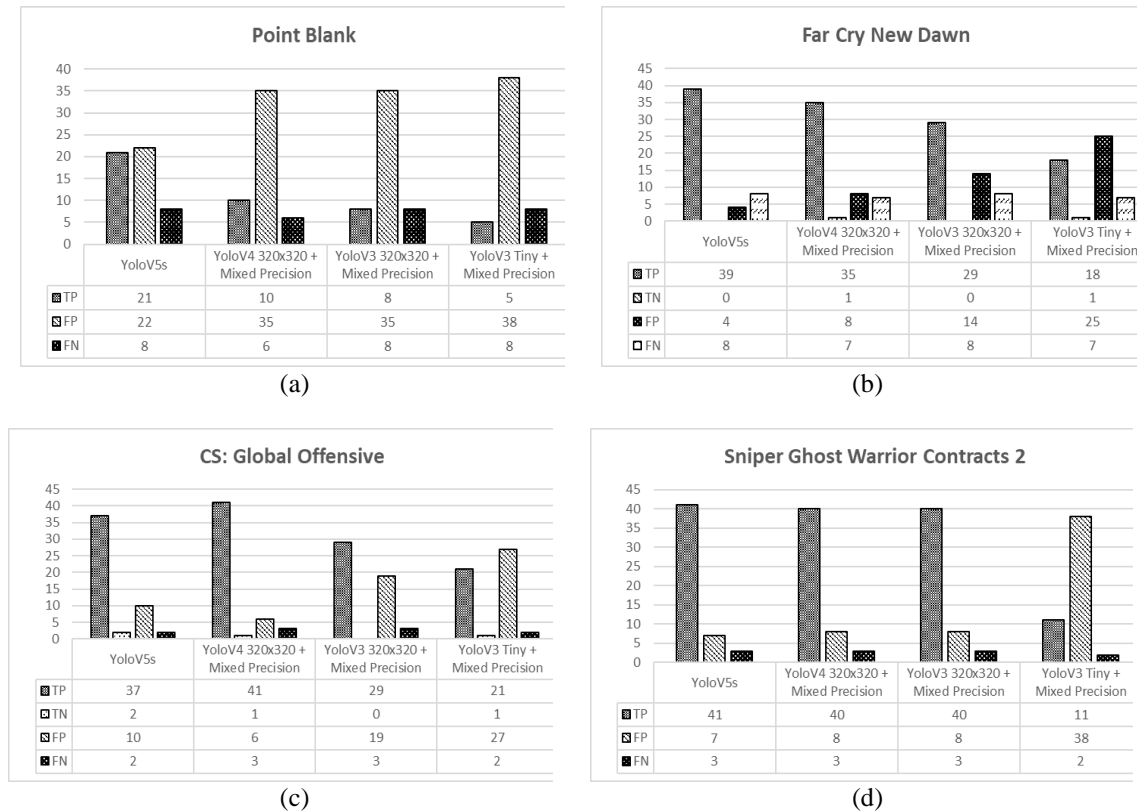


Figure 12. Experiment test results on several games using different YOLOV4 versions; (a) game Point Blank obtained, (b) game Counter Strike, (c) game Far Cry New Dawn, and (d) game Sniper Ghost Warrior Contracts 2

Table 2. Comparison of FPS from experiment using different YOLO versions and central processing unit (CPU,) GPU spectral-level parallelism (SP), and GPU high-performance (HP)

Algorithm	CPU	GPU (single precision)	GPU (half precision)
YOLOv3 tiny	6.5 FPS	15.4 FPS	32 FPS
YOLOv3 320x320	8.5 FPS	12.6 FPS	18.13 FPS
YOLOv4 tiny	6.5 FPS	15.2 FPS	35 FPS
YOLOv4 320x320	8.5 FPS	12.4 FPS	18.32 FPS
YOLOv5s	10 FPS	29 FPS	34 FPS
YOLOv6s	12 FPS	30 FPS	36.5 FPS

Further, we also experimented using YOLOV6 and YOLOV5. We found that the YOLOV6 has a higher FPS (36.5) than YOLOV5 (34). Meanwhile, the FPS of the other versions of YOLO is not excessively high, mainly if the system is operated through the CPU, not the GPU. The YOLOV4 presents sufficiently high FPS but low accuracy, inhibiting its implementation in the real-time game. Our findings show that better accuracy in one FPS game is not necessarily better in another FPS game. The proposed method may get different results from specific game conditions such as environmental conditions (light, dark, distance of the detected object person, texture and detail of the object person, and conditions of background complexity). This research comprehensively investigates YOLO's computing time and accuracy in several FPS games and with different computing methods. Additional and in-depth research is needed to confirm computing times on several different device specifications and trials on the latest version of YOLO to confirm better computing cost efficiency.

5. CONCLUSION

The YOLO experiment presents relatively high accuracy and framerate, as confirmed by the results of experiments and tests on several games. However, the obtained results are highly influenced by the computer specification. In our experiment, we conducted person detection through screenshots on every

frame. Further, the screenshot algorithm affects the computation speed. Our experiment results suggested that PyAutoGUI is not recommended due to its excessively high computation, which is less applicable for FPS games with a high amount of frame per second. The results also showed that YOLOV5 has the highest accuracy detection than the previous version of YOLO (YOLOV4 and YOLOV3). In detail, the highest accuracy has been obtained from the experiment using YOLOV5 on Far Cry New Dawn (92%), A Sniper Ghost Warrior Contracts 2 (86%), Counter Strike Global Offensive (76%), and Point Blank (57%). Besides, YOLOV5s also presents a higher framerate with 34 FPS carried out using half precision GPU.

Similarly, YOLOv4 tiny also has a relatively high frame rate but produces lower accuracy than YOLOV5. Further, we also conducted an experiment using YOLOV6s, resulting in a slightly higher frame rate of 36 FPS. Therefore, the detection accuracy is impacted by the object's differences: body and the object-viewpoint distance. Lastly, the detection is less accurate when the object is small.

From the test results, YOLOV5s or even YOLOV6s is quite good when used as an aimbot. The value of 36.5 fps is enough to give users room to play well-tested games, but they must use a half-precision GPU. It also needs to be taken into consideration that nowadays, many games are starting to run at around 60 FPS, so if one uses it to play multiplayer, it will be late. As of when this article was written, YOLOV8 has been published; the author is also confident that with the newer YOLO, computing costs will also be lower, even though testing for the aimbot has yet to be carried out. Apart from aimbots, in the future, this proposed research can also be used as real-time streaming object detection on the camera that captures the monitor screen, on CCTV installed in buildings, traffic, traffic lights, toll roads, or security system CCTV.

REFERENCES




- [1] H. K. Rhee, D. H. Song, and J. H. Kim, "Comparative analysis of first person shooter games on game modes and weapons – military-themed, overwatch, and player unknowns' battleground," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 1, pp. 116-122, Jan. 2019, doi: 10.11591/ijeecs.v13.i1.pp116-122.
- [2] D. H. Song, H. K. Rhee, and J. H. Kim, "Gender stereotype and hostile sexism among young Korean gamers based on teammate selection strategy and game style preferences," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 19, no. 3, pp. 1512–1518, Sep. 2020, doi: 10.11591/ijeecs.v19.i3.pp1512-1518.
- [3] A. E. and A. Rollings, "Fundamentals of game design," *Choice Reviews Online*, vol. 47, no. 08, pp. 47-4462-47-4462, 2010, doi: 10.5860/choice.47-4462.
- [4] M. J. P. Wolf and B. Perron, "The routledge companion to video game studies," *The Routledge Companion to Video Game Studies*, p. 544, 2014, doi: 10.4324/9780203114261.
- [5] S. Schneider, "The 5 Best 'Doom' Clones Ever Released | Tech Times," *Tech Times*, 2016.
- [6] F. Cifaldi, "The Gamasutra Quantum Leap Awards: First-Person Shooters," *Game Developer*, 2006.
- [7] R. Weber, K. M. Behr, R. Tamborini, U. Ritterfeld, and K. Mathiak, "What do we really know about first-person-shooter games? An event-related, high-resolution content analysis," *Journal of Computer-Mediated Communication*, vol. 14, no. 4, pp. 1016–1037, Jul. 2009, doi: 10.1111/j.1083-6101.2009.01479.x.
- [8] C. S. Lee, H. K. Kim, H. R. Won, and K. Kim, "A method for preventing online games hacking using memory monitoring," *Electronics and Telecommunications Research Institute*, vol. 43, no. 1, pp. 141–151, Feb. 2021, doi: 10.4218/etrij.2019-0427.
- [9] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 779–788, 2016, doi: 10.1109/CVPR.2016.91.
- [10] R. A. Asmara, B. Syahputro, D. Supriyanto, and A. N. Handayani, "Prediction of traffic density using yolo object detection and implemented in raspberry pi 3b+and intel NCS 2," in *4th International Conference on Vocational Education and Training*, IEEE, Sep. 2020, pp. 391–395, doi: 10.1109/ICOVET50258.2020.9230145.
- [11] K. A. Adeniji, M. O. Onibonjo, A. Minevesho, T. Ejidokun, and O. O. Omitola, "A robust 4.0 dual-classifier for determining the internal condition of watermelons using YOLOv4-tiny and sensory," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 28, no. 3, pp. 1834–1844, Dec. 2022, doi: 10.11591/ijeecs.v28.i3.pp1834-1844.
- [12] H. S. Abdul-Ameer, H. J. Hassan, and S. H. Abdullah, "Development smart eyeglasses for visually impaired people based on you only look once," *Telecommunication Computing Electronics and Control*, vol. 20, no. 1, pp. 109–117, Feb. 2022, doi: 10.12928/TELKOMNIKA.v20i1.22457.
- [13] K. Gayathri and S. Thangavelu, "Novel deep learning model for vehicle and pothole detection," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 3, pp. 1576–1582, Sep. 2021, doi: 10.11591/ijeecs.v23.i3.pp1576-1582.
- [14] M. A. Anwer, S. M. Shareef, and A. M. Ali, "Accident vehicle types classification: A comparative study between different deep learning models," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 21, no. 3, pp. 1474–1484, Mar. 2021, doi: 10.11591/ijeecs.v21.i3.pp1474-1484.
- [15] A. Hanafi, L. Elaachak, and M. Bouhorma, "Machine learning based augmented reality for improved learning application through object detection algorithms," *International Journal of Electrical and Computer Engineering*, vol. 13, no. 2, pp. 1724–1733, Apr. 2023, doi: 10.11591/ijece.v13i2.pp1724-1733.
- [16] H. F. Al-Selwi, N. Hassan, H. B. A. Ghani, N. A. B. A. Hamzah, and A. B. A. Aziz, "Face mask detection and counting using you only look once algorithm with Jetson Nano and NVIDIA giga texel shader extreme," *International Journal of Artificial Intelligence*, vol. 12, no. 3, pp. 1169–1177, 2023, doi: 10.11591/ijai.v12.i3.pp1169-1177.
- [17] M. A. Benallal and M. S. Tayeb, "An image-based convolutional neural network system for road defects detection," *International Journal of Artificial Intelligence*, vol. 12, no. 2, pp. 577–584, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp577-584.
- [18] A. Raj and R. Sugumar, "Estimating social distance in public places for COVID-19 protocol using region CNN," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 1, pp. 414–421, Apr. 2023, doi: 10.11591/ijeecs.v30.i1.pp414-421.
- [19] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv e-prints*, 2018.
- [20] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv*, 2020, doi: 10.48550/arXiv.2004.10934.
- [21] X. Zhu, S. Lyu, X. Wang, and Q. Zhao, "TPH-YOLOv5: Improved YOLOv5 Based on Transformer Prediction Head for Object

- Detection on Drone-captured Scenarios,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2021-Octob, pp. 2778–2788, 2021, doi: 10.1109/ICCVW54120.2021.00312.
- [22] C. Li *et al.*, “YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications,” *arXiv*, 2022, doi: 10.48550/arXiv.2209.02976.
- [23] S. F. Yeung, J. C. S. Lui, J. Liu, and J. Yan, “Detecting cheaters for multiplayer games: Theory, design and implementation,” in *2006 3rd IEEE Consumer Communications and Networking Conference*, IEEE, 2006, pp. 1178–1182, doi: 10.1109/CCNC.2006.1593224.
- [24] J. Yan and B. Randell, “A systematic classification of cheating in online games,” *Proceedings of 4th ACM SIGCOMM Workshop on Network and System Support for Games*, pp. 1–9, 2005, doi: 10.1145/1103599.1103607.
- [25] X. B. Shi, X. X. Zhou, F. Liu, and Y. Wang, “A cheat-detection method based on fuzzy synthesis decision for aimbot cheating,” in *International Conference on Internet Technology and Applications*, IEEE, Aug. 2010, pp. 1–4, doi: 10.1109/ITAPP.2010.5566408.
- [26] S. Y. Yu, N. Hammerla, J. Yan, and P. Andras, “A statistical aimbot detection method for online FPS games,” in *Proceedings of the International Joint Conference on Neural Networks*, IEEE, Jun. 2012, pp. 1–8, doi: 10.1109/IJCNN.2012.6252489.
- [27] E. Lee, J. Woo, H. Kim, A. Mohaisen, and H. K. Kim, “You Are a Game Bot!: Uncovering Game Bots in MMORPGs via Self-similarity in the Wild,” in *23rd Annual Network and Distributed System Security Symposium*, Reston, VA: Internet Society, 2016, doi: 10.14722/ndss.2016.23436.
- [28] A. Kanervisto, T. Kinnunen, and V. Hautamaki, “GAN-Aimbots: Using Machine Learning for Cheating in First Person Shooters,” *IEEE Transactions on Games*, vol. 15, no. 4, pp. 566–579, Dec. 2023, doi: 10.1109/TG.2022.3173450.
- [29] J. P. Pinto, A. Pimenta, and P. Novais, “Deep Learning and Multivariate Time Series for Cheat Detection in Video Games,” in *2021 IEEE 8th International Conference on Data Science and Advanced Analytics*, IEEE, Oct. 2021, pp. 1–2, doi: 10.1109/DSAA53316.2021.9564219.
- [30] R. Brunelli, “Template Matching Techniques in,” *Theory and Practice*, no. May, 2009.
- [31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *Proceedings of the IEEE International Conference on Computer Vision*, IEEE, Nov. 2011, pp. 2564–2571, doi: 10.1109/ICCV.2011.6126544.
- [32] D. G. Lowe, “Object recognition from local scale-invariant features,” in *Proceedings of the IEEE International Conference on Computer Vision*, IEEE, 1999, pp. 1150–1157, doi: 10.1109/iccv.1999.790410.
- [33] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS, pp. 404–417, 2006, doi: 10.1007/11744023_32.
- [34] R. K. McConnell, “Method of and apparatus for pattern recognition,” *Wayland Research Inc., Wayland, Mass.*, vol. 96, no. 19, pp. 62–66, 1986.
- [35] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2001, pp. 511–518, doi: 10.1109/cvpr.2001.990517.
- [36] K. Arai and R. Andrie, “Gait recognition method based on wavelet transformation and its evaluation with Chinese Academy of Sciences (CASIA) gait database as a human gait recognition dataset,” *Proceedings of the 9th International Conference on Information Technology*, pp. 656–661, 2012, doi: 10.1109/ITNG.2012.164.
- [37] R. A. Asmara, M. Ridwan, and G. Budiprasetyo, “Haar Cascade and Convolutional Neural Network Face Detection in Client-Side for Cloud Computing Face Recognition,” in *Proceedings 1st International Conference on Electrical and Information Technology*, IEEE, Sep. 2021, pp. 1–5, doi: 10.1109/IEIT53149.2021.9587388.
- [38] Y. T. Chan, K. C. Ho, and S. K. Wong, “Aircraft identification from RCS measurement using an orthogonal transform,” *IEEE Proceedings: Radar, Sonar and Navigation*, vol. 147, no. 2, pp. 93–102, 2000, doi: 10.1049/ip-rsn:20000240.
- [39] G. S. Cox, “Review: Template Matching and Measures of Match in Image Processing,” *University of Cape Town, Department of Electrical Engineering, South Africa, Technical report.*, pp. 1–24, 1995.
- [40] L. Di Stefano and S. Mattoccia, “Fast template matching using bounded partial correlation,” *Machine Vision and Applications*, vol. 13, no. 4, pp. 213–221, Feb. 2001, doi: 10.1007/s00138-002-0070-5.
- [41] L. Di Stefano, S. Mattoccia, and M. Mola, “An efficient algorithm for exhaustive template matching based on normalized cross correlation,” in *Proceedings-12th International Conference on Image Analysis and Processing*, IEEE Comput. Soc., 2003, pp. 322–327, doi: 10.1109/ICIAP.2003.1234070.
- [42] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, “Libra R-CNN: Towards balanced learning for object detection,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019, pp. 821–830, 2019, doi: 10.1109/CVPR.2019.00091.
- [43] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017, doi: 10.1109/TPAMI.2016.2577031.
- [44] R. Girshick, “Fast R-CNN,” *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.
- [45] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.
- [46] W. Liu *et al.*, “SSD: Single shot multibox detector,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9905 LNCS, pp. 21–37, 2016, doi: 10.1007/978-3-319-46448-0_2.
- [47] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal Loss for Dense Object Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, Feb. 2020, doi: 10.1109/TPAMI.2018.2858826.
- [48] X. Zhu, H. Hu, S. Lin, and J. Dai, “Deformable convnets V2: More deformable, better results,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019, pp. 9300–9308, 2019, doi: 10.1109/CVPR.2019.00953.
- [49] J. Dai *et al.*, “Deformable Convolutional Networks,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 764–773, 2017, doi: 10.1109/ICCV.2017.89.
- [50] S. Zhang, L. Wen, Z. Lei, and S. Z. Li, “RefineDet++: Single-Shot Refinement Neural Network for Object Detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 2, pp. 674–687, 2021, doi: 10.1109/TCSVT.2020.2986402.
- [51] E. Evans-Thirlwell, “The history of the first person shooter | PC Gamer,” *PCGamer.com*, 2017.
- [52] F. Leroux, “The best FPS games on Android in 2022,” *AndroidPolice.com*, 2022.




- [53] X. Lan, Y. Zhang, and P. Xu, "An Overview on Game Cheating and Its Counter-measures," 2009.
- [54] W. C. Feng, E. Kaiser, and T. Schluessler, "Stealth measurements for cheat detection in on-line games," in *Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games*, New York, NY, USA: ACM, Oct. 2008, pp. 15–20, doi: 10.1145/1517494.1517497.
- [55] M. Jang, H. Kim, and Y. Yun, "Detection of DLL Inserted by Windows Malicious Code," in *2007 International Conference on Convergence Information Technology*, IEEE, Nov. 2008, pp. 1059–1064, doi: 10.1109/iccit.2007.320.
- [56] Z. Xu, S. Ray, P. Subramanyan, and S. Malik, "Malware detection using machine learning based analysis of virtual memory access patterns," in *Proceedings of the 2017 Design, Automation and Test in Europe*, IEEE, Mar. 2017, pp. 169–174, doi: 10.23919/DAT.2017.7926977.
- [57] T. Witschel and C. Wressnegger, "Aim low, shoot high: Evading aimbot detectors by mimicking user behavior," *Proceedings of the 13th European Workshop on Systems Security*, pp. 19–24, 2020, doi: 10.1145/3380786.3391397.
- [58] M. Willman, "Machine Learning to identify cheaters in online games," *Proceedings of the 13th European workshop on Systems Security*, vol. 44, 2020.
- [59] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.
- [60] "Release v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference · ultralytics/yolov5 · GitHub".
- [61] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *arXiv*, pp. 7464–7475, 2023, doi: 10.1109/cvpr52729.2023.00721.
- [62] G. Jocher, "YOLOv5 Focus() Layer Discussion #3181 ultralytics/yolov5 GitHub," *GitHub*.
- [63] M. Schoentgen, "GitHub - BoboTiG/python-mss: An ultra fast cross-platform multiple screenshots module in pure Python using ctypes.," *GitHub*, 2022.
- [64] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693, pp. 740–755, 2014, doi: 10.1007/978-3-319-10602-1_48.

BIOGRAPHIES OF AUTHORS






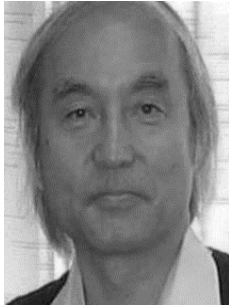
Rosa Andrie Asmara    (Fellow, IEEE) received the B.E. degree in Electronics Engineering from Brawijaya University, Malang, Indonesia, and the M.S. degree in Multimedia Engineering, from Institute of Technology Sepuluh Nopember, Surabaya, Indonesia, in 2004 and 2009, respectively. Graduated from Saga University, Japan from Department of Computer Science in 2013 and become post doctoral researcher in System Creation Laboratorium until 2014. Currently a lecturer in the Informatics Engineering study program, Department of Information Technology at State Polytechnics of Malang, Indonesia, started from 2005. Over 15 years on the institutions, have principally taught courses dealing with computer architecture and organizations, operating systems, computer networking, programming and algorithms, database, web programming, image processing, computer vision, and computer graphics. Research interests include signal processing, image processing, artificial intelligence, and computer vision. He can be contacted at email: rosa.andrie@polinema.ac.id.






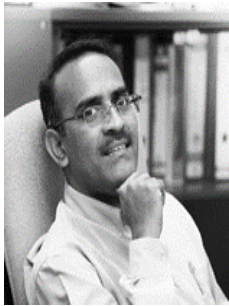
Muhammad Rahmat Samudra Anugrah    B.E. student in Informatics Engineering Study Program, State Polytechnic of Malang, Indonesia. Currently joining in Computer Vision Laboratory and have some research dealing with object detection in First Person Shooting Games. He can be contacted at email: syllxwestern@gmail.com.






Dimas Wahyu Wibowo    received the B.E. degree in Electronics Engineering from Brawijaya University, Malang, Indonesia, and the M.S. degree in Electronics Engineering, from Brawijaya University, Malang, Indonesia, in 2008 and 2014, Currently a lecturer in the Informatics engineering field, Department of Electrical Engineering at State Polytechnics of Malang, Indonesia, started from 2015. Over 8 years on the institutions, have principally taught courses dealing with computer architecture and organizations, operating systems, programming and algorithms, database, web programming and game programing. research interests include multimedia, virtual reality, augmented reality, artificial intelligence, and game. He can be contacted at email: dimas.wahyu@polinema.ac.id.






Kohei Arai    received B.S., M.S., and Ph.D. degrees in 1972, 1974 and 1982, respectively. He was with The Institute for Industrial Science and Technology of the University of Tokyo from April 1974 to December 1978 and was with National Space Development Agency of Japan from January 1979 to March 1990. During 1985 to 1987, he was with Canada Centre for Remote Sensing as a Post Doctoral Fellow of National Science and Engineering Research Council of Canada. He moved to Saga University as a Professor in the Department of Information Science in April 1990. He was a councilor for the Aeronautics and Space related to the Technology Committee of the Ministry of Science and Technology during 1998 to 2000. He was a councilor of Saga University for 2002 and 2003. He also was an executive councilor for the Remote Sensing Society of Japan for 2003 to 2005. He has been an Adjunct Professor of University of Arizona, USA since 1998. He can be contacted at email: arai@cc.saga-u.ac.jp.






Mohd Aboobaidur Burhanuddin    received Diploma, B.Sc., M.Sc., and Ph.D. degrees in 1994, 2002, 2004, and 2009 respectively. Currently, is a Senior Lecturer and Head of Department for the Industrial Computing Department. He is a lecturer in soft computing and mathematics, teaching theory and practical courses for post-graduates and undergraduates students. After a few years working experience in the multi-national industries, includes Rubber Industrial Smallholder Development Authority, Intel Technology and ESSO Production Incorporation, he successfully completed his M.Sc. and Ph.D. With good experience in industry and manufacturing lines, Burhanuddin has sought to bring together disciplines, computer and mathematics to obtain M.Sc. and Ph.D. He joined Universiti Teknikal Malaysia Melaka at Faculty of Information and Communication Technology in 2004. Then he joined in King Abdulaziz University, Rabigh, Kingdom of Saudi Arabia as a secondment program for 2 years. He has been active at all levels of the university development and one of the pioneers in building the faculties in Universiti Teknikal Malaysia Melaka and King Abdulaziz University Rabigh. He can be contacted at email: burhanuddin@utem.edu.my.



Anik Nur Handayani    (Member, IEEE) received the B.E. degree in Electronics Engineering from Brawijaya University, and the M.S. degree in Electrical Engineering, from Institute of Technology Sepuluh Nopember Surabaya, Indonesia, in 2004 and 2008. She was graduate from Computer Science Doctoral degree in Saga University Japan in 2014. She can be contacted at email: aniknur.ft@um.ac.id.



Farradila Ayu Damayanti    B.E. degree in Informatics Engineering from State Polytechnic of Malang. Currently pursuing a master's program in Information Technology Engineering from State Polytechnic of Malang, Indonesia. She can be contacted at email: farradila429@gmail.com.